August 1982 LIDS-R-1227

TECCNET:

A Testbed for Evaluating

Command and Control NETworks

by

Elizabeth R. Ducot

ABSTRACT

TECCNET (Testbed for Evaluating Command and Control NETworks) is a small, expandable software system created to support C3 system research. It has been designed 1) to highlight the complex interactions between the distributed command and control network elements, the information flow network and the environment within which the systems function, and 2) to support the development of an Information Intermediary between the C3 Network and the User. TECCNET is interactive and accommodates three basic user activities: definition of a model to simulated, generation of a scenario, and execution of an experiment. An initial modeling environment has been specified to simulate the management of the network. The algorithm used to demonstrate the system is one proposed by Golestaani as part of his PhD research, which treats flow control and routing together within a unified framework.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
AUG 1982		**00-08-1982 to 00-08-1982**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
TECCNET: A Testbed fro Evaluating Command and Control NETworks	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Massachusetts Institute of Technology,Laboratory for Information and Decision Systems,77 Massachusetts Avenue,Cambridge,MA,02139-4307**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**72**	

ACKNOWLEDGMENTS

This research has benefited substantially from the interactions with many individuals in different fields. Special acknowledgments are due to Professors Robert Gallager, Dimitri Bertsekas and Pierre Humblet of the MIT Laboratory for Information and Decision Systems for their willingness to describe not only the algorithms, but also the current approaches and developments in computer communication networks. The contribution of Mr. Eli Gafni in exploring some of the specifics of the routing/flow control algorithm is also acknowledged.

Professor Wilbur B. Davenport, Jr. (formerly of MIT and now at the University of Hawaii) has supplied valuable information on the communications issues associated with distributed data base systems.

Thanks also go to Mr. Richard Marcus, MIT/LIDS, who has provided some of the ideas and tools used in the development of the Conversational Interface.

And finally, my appreciation to Dr. Alexander H. Levis for the many opportunities to discuss the overall TECCNET design, and its application to C3 research problems.

This work was conducted at the Laboratory for Information and Decision Systems at MIT, with support extended by the Air Force Office of Scientific Research under contract No. AFOSR-80-0229.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

1.1 INFORMATION FLOW IN C3 SYSTEMS

The need to limit the flow of information in a Command, Control, and Communications (C3) system operating under stress, while preserving its effectiveness in the context of changing military situations, has become increasingly critical. It is this need that has provided the motivation for both the research activity and the development of the research support system described in the report that follows.

1.1.1 Backgound

In this work, the C3 system is visualized as an information flow network. This description encompasses not only the communications systems that transmit data and messages, but also the processing and storage systems that acquire, translate, manipulate, and disseminate information. The performance of this network may be described (at least conceptually) in terms of its ability to deliver, at designated points, the desired information so that, upon arrival, it is timely, accurate, complete, and easy to use.

It is clear that the underlying C3 system problem is extremely complex. The system elements that comprise the information flow network are highly distributed, have diverse physical characteristics, and are often governed by ill-defined

operational constraints and procedures. The technologies that affect the elements of this system are changing rapidly; advances in electronic weaponry, sensors, and computers, for example, combine with changes in the way information is used by the commander to increase both the flow of information in the network and the time pressures associated with its delivery.

Even under somewhat benign conditions, the task of supporting this flow of information is a formidable one. However, when the tactical situation intensifies, the load on the system increases substantially, just when the external stress on the network induced by a hostile atmosphere is at its peak. Competition for the same resources to move, process, store, and display information also intensifies -- with frequently disastrous results (e.g. excessive message delays, or system and user information overloads). Thus, the C3 system, viewed in terms of how well it provides the information support expected by the decision-maker, is perceived to degrade exactly when it is most important that it operate well: when battle information is flowing and the time available for decision making is short. It follows, then that there is need to modify the information flow to match it in real-time to the facilities and the time available for processing.

1.1.2 The Information Intermediary and the TECCNET System

In formulating the research problem, an approach is taken that seeks to blend the needs of the users with the

capabilites of the system. This approach is predicated on the notion that the development of successful information flow control techniques must somehow exploit the interrelationship between the activities of the user and those of the network. As this approach was refined, it became clear that there was need for a framework in which these interactions could be expressed and analyzed.

In designing the appropriate flow control techniques, one must take into account the following: the decision-maker alters the way in which he approaches problems, depending on his awareness of, and belief in, the potential support to be obtained from his information flow network. If one attempts to characterize the commander's reliance on real-time information, a dilemma is apparent; a dilemma perhaps best illustrated by the following quotation drawn from a discussion on the evolution of battle concepts:

> "What he needs to know depends to some extent on what he is attempting to accomplish. What he attempts to accomplish is guided by his knowledge of what information can be provided..."
>
> Major General Jasper A. Welch [WEL80]

In the case of the information flow network, the statement "can be provided" should be interpreted to include not only what is available, but also "by what time" and "at what cost". Moreover, it should be noted that the notion of cost intended here extends beyond the point of view of the individual user, to include the community of users affected by his actions. Since the attributes

of availability, timeliness and cost may be as volatile as the user's information reqirements, some form of "negotation for service" between the user and the information system is needed.

From the research point of view, the introduction of the notion of flow control for information, as opposed to data, in the network, requires the development of an Information Intermediary between the user and the network. This intermediary must have access to a specially developed model of the information network; a model that must incorporate principles drawn from many current areas of research. The development of the local status model, involving network, data base, and user information, requires the flow of control information throughout the system. It is conceivable that unless extreme care is exercised in the design, (i.e., drawing on existing network quantities already available and exploiting local data collected at the nodes), this flow could become prohibitive, making the intermediary an undesirable drain on the system resources.

The complexity of the modeling problem is illustrated by the following example describing a simple request for information made by the user. Current technologies for managing information tend to create numerous processing layers between the user desiring information service and the physical components of the system that provide that service. A simple question passes, in general, first through the user/system interface where it may be expanded into a more complex series of data base queries appropriate for a given model of the data. The data base system processes the queries, relying on the communication network to

interconnect its various processors which may be physically or logically distributed. For simplicity, processing at each layer may operate somewhat independently of activities of layers above, layers below, or elements elsewhere in the system. An intermediate layer, for example, may predicate its processing on an assumption that "perfect processing" can be obtained from the layer below, and that demands from the layer above must be met as received. While this tends to minimize the interdependency of processing layers and hence, the amount of detailed control information passed between them, it is apparent that large volumes of low level data may be moved through the information flow network without regard for its current condition. As a result, by the time the processed answer has reached the user, it may be of diminished importance. Moreover, since the processing at each of the layers operates independently, the fact that a descriptive aggregate response (one that could have been delivered sooner at less cost) may have been adequate under the circumstances will not be recognized.

The alternative to this layered approach applied to the management of information flow networks is not obvious. However, if the intermediary is to act as an intelligent mechanism for the control of the information flow, a vertical link between the natural processing layers as well as horizontal links between the network elements, must be created. The following key functions must be performed:

1) monitor the network to determine system reponse, conditioned on the environment and the user request issued;

2) assist the user to reformulate his request in light of network conditions.

The implication of establishing the vertical link between processing layers is that the traditional boundaries between research disciplines must some how be bridged. Moreover, this must be accomplished in such a way that the exploration of the subtle interactions between the system elements, and the various models, algorithms and procedures that characterize each layer is encouraged. Numerous analytically based approaches to portions of the information flow problem are under development by researchers at the MIT Laboratory for Information and Decision Systems [HUA82],[TEN82]. While these developments appear promising, it has been recognized that if these research efforts are to be extended to address interactions between the various information processes, a consistent framework for analysis across research areas must be created. (1) In addition, if these extensions are to contribute toward the development of the flow control techniques for the intermediary, this framework must be one that supports both experimentation and model development.

In order to provide such a framework, consistent with the research issues raised above, the development of a small expandable software system was required. The Information Flow Network Testbed (TECCNET: Testbed for Evaluating Command and Control NETworks), and its design, implementation and use, are the subject of the report that follows.

---

(1) Personal communication, Professors W.B.Davenport, Jr. and R.R.Tenney

1.2 THE REPORT IN OUTLINE

Section II provides an overview of the TECCNET system, indicating the goals to be met and a number of the software design objectives that were established for the project. The design and implementation considerations, as they applied to the selection of a software development environment, are presented; and the structure of TECCNET is described. Some detail on each of the TECCNET modules is given, suitable for users interested in either analysis or model and algorithmic development.

In Section III, the initial algorithm, implemented within the TECCNET framework, is outlined. The formulation of this algorithm, integrating flow control and routing for data communicaton networks, is presented; and the considerations surrounding its distributed implementation are indicated. Two test situations are described, suggesting the usefulness of TECCNET as a research tool.

The conclusions are presented in Section IV.

II. THE TECCNET SYSTEM

2.1 OVERVIEW OF THE SYSTEM

The motivation for the development of an experimental laboratory to support the C3 research efforts at MIT was indicated in the previous section. TECCNET was designed in response to this need, and, beginning in March 1981, a skeleton system was implemented.

The goals of this effort were the following: The first was to promote the integration of ideas from several research areas (e.g, distributed data base, sensor and network management, information processing and presentation) within a consistent framework. The second was to improve our understanding, through experimentation, of the complex interactions between the distributed command and control network elements, the information flow network itself, and the environment within which the systems function. The third, and most important, was to provide a facility for the development of the Information Intermediary.

In order that TECCNET meet these goals and become an effective vehicle for our C3 research, a number of design objectives were established.

1) The testbed should be structured so that it meets the needs of users with different levels of software and system expertise. System interface and support software should be provided to facilitate both modeling and testing activities.

2) Default models and representations of the system should be available in order to reduce the effort required to initiate simple experiments.

3) The software system should be small, with a controlled plan for expansion, so that it will remain a manageable tool for project research.

4) The modeling tools available within TECCNET should provide the capability for representing the asynchronous interactions and complex protocols that are characteristic of the models and algorithms likely to be explored in the near future.

The software specifications implied by the preceding statements encompass a broad range of system functions, in addition to those ordinarily associated with simulation and analysis. As a result, substantially different design techniques were employed in the development of TECCNET from those customarily applied to the creation of software for algorithmic and system research. Indeed, the final design of TECCNET was influenced strongly by the principles applied to the development of special-purpose computer operating systems. Projects of this type, with the focus on a comprehensive user/system interface, are often very ambitious undertakings. It was clear that if such an approach was to be pursued on a modest scale, its success would be critically dependent on the selection of a software development environment. The issues surrounding both the choice of a computer system and system development language are outlined in the section that follows.

2.1.1 Design and Implementation Considerations

The MIT Multics System (1) was selected to host the testbed for a variety of reasons. It is one of several multiple user computer facilities with which MIT researchers and students are both familiar and comfortable. Moreover, the extensive Multics network, the links throughout the campus, via dial-up lines and over networks such as the ARPA net, make Multics practically, as well as physically, accessible to potential users of TECCNET.

A more fundamental reason for the selection, from the point of view of the design of a research tool such as TECCNET, was derived from the design of Multics itself. The Multics operating system is layered in such a way that the development of user-oriented subsystems such as TECCNET is encouraged. As a general policy PL/I, a language encompassing many of the characteristics common to problem-oriented languages, was used whenever possible as the system programming language [COR69]. Tools for controlling the resources of the system and sophisticated mechanisms for defining the interface between the user and the system were developed early in the implementation of the Multics system. These tools were used in a bootstrap fashion to create much of the basic system software, an approach widely in use today. These same mechanisms are still in place and are available to the user who wishes to bypass the standard Multics

(1) Multics stands for MULTiple Information and Computing Service and is the result of a joint research effort between researchers at MIT and Honeywell. [COR72]

facilities and to access the native internal programming environment of Multics (consisting of a stack-oriented, pure-procedure, collection of PL/I procedures). Such users, or subsystem developers, essentially operate one level deeper within the system than the majority of users with no loss of service. In this implementation environment, a subsystem developer, wishing to know the status of physical processes related to his activities, is provided with the critical system data structures that are normally highly protected. Moreover, through the use of PL/I, one can create direct access to Multics system routines that control not only internal processes but peripheral devices as well. These features permit one not only to gain considerable execution-time control over the system resources, but to do so from within a general purpose implementation language. Therefore, one may, in an efficient manner, develop self-contained system executives that can perform a broad range of functions tailored to particular users and/or applications.

These characteristics of the Multics environment were exploited in the design of TECCNET. As a result, the system support software (such as terminal, interactive data and file handling routines) were created for TECCNET with a relatively small investment in both time and effort. PL/I has been used to implement the basic TECCNET system. For reasons indicated above, the use of other languages for the executive portion of TECCNET, although feasible, was not considered seriously. Only in PL/I could the desired system data structure be replicated readily and the data required for the system calls be generated in their

natural format.

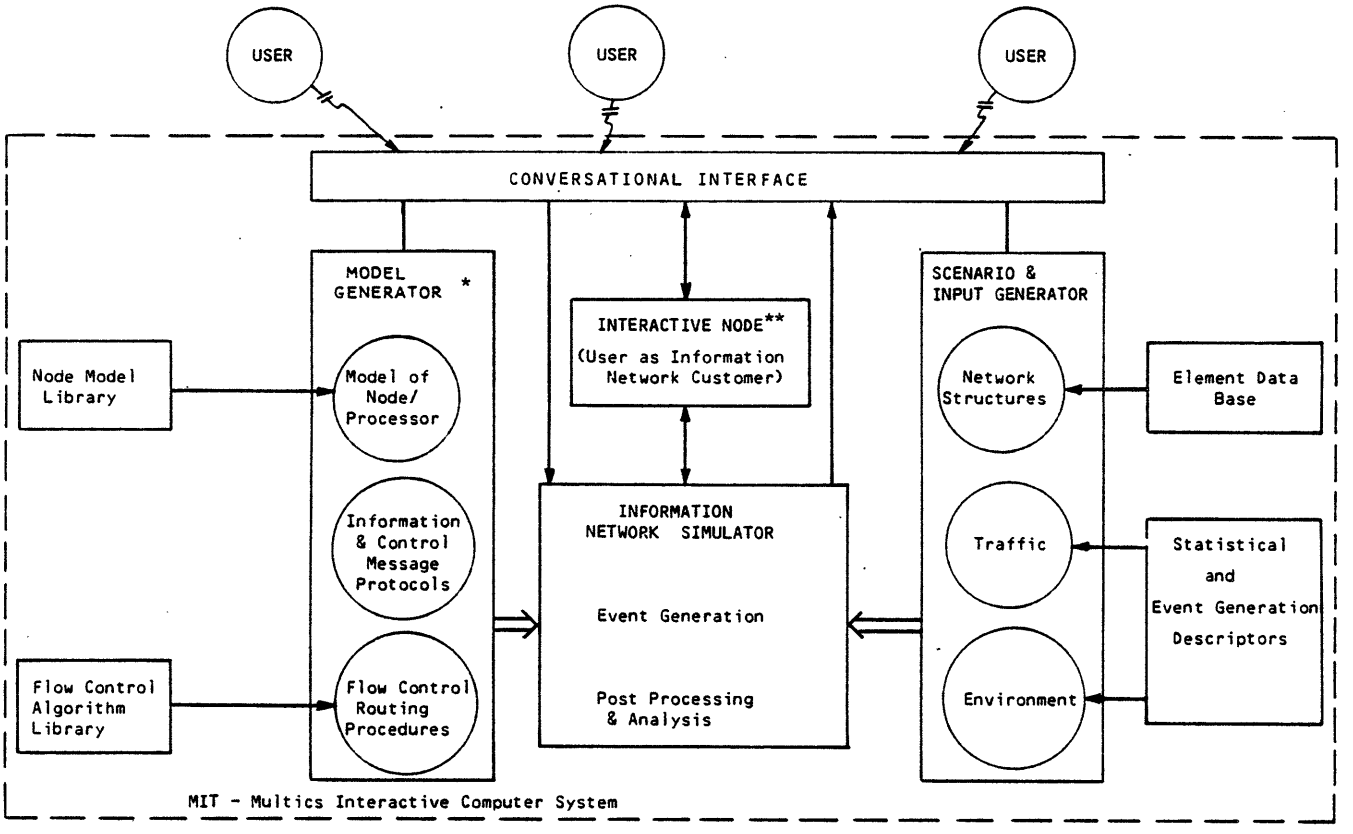
The selection of a language for algorithmic as opposed to system development was not governed by these same considerations, and therefore was not so straightforward. A number of languages can be used within the PL/I system environment, and algorithms implemented in a variety of ways could be made compatible with the TECCNET system. Ultimately, the choice of PL/I for the initial implementation within TECCNET rested on a number of subjective considerations: 1) the features of PL/I seemed well-matched to the requirements for pointer and character manipulation and list processing for the class of algorithms to be considered, 2) PL/I was a stable language system on Multics (both the FORTRAN and PASCAL compilers were undergoing substantial upgrades during the design phase of TECCNET), and 3) future development and enhancements would not be not constrained by the selection of PL/I; modules developed in PASCAL and FORTRAN, for example, could be encorporated later if desired.

The actual structure of the TECCNET system, its system executive and the underlying modules were designed to operate within this system environment. They are described briefly below, and in greater detail in the sections that follow.

2.1.2 Structure of TECCNET

The TECCNET system is interactive. Communication between the user and the TECCNET system takes place through the Conversational Interface. The organization of the testbed

beneath the Interface is by blocks separated into basic user activities, or functions, and structured according the diagram in Figure 1.



*Initial version allows selection of only one process model (simple input/output store & forward node) and one flow control algorithm with baseline message set.

**Planned for future development.

Figure 1 Structure of TECCNET

The first of these functions is the specification of the modeling environment. This permits the user to define his view of the C3 information flow network by combining models of the local processing nodes, constraints on the movement of messages and the algorithms for managing the communication

network. A library structure has been developed to house these modeling and algorithmic building blocks; descriptive information is associated with each entry in the library.

The second function is a data-intensive one in which the user defines the conditions to be simulated. Three steps are required:

1) the specification of the network topology, capacity of the links etc.,

2) the association of nodes with particular processing models and descriptions of the traffic between them, and

3) the representation of the environment.

The data are organized for permanent storage and cataloged for the user.

The third function is the execution of the experiment based on the combined model and scenario. The execution module relies on discrete event simulation techniques in order to capture the asynchronous behavior of the interactions between the distributed models and algorithms.

2.2 THE CONVERSATIONAL INTERFACE

As depicted in Figure 1, the Conversational Interface provides the link between the user and the body of the TECCNET system. Communication is interactive, with commands and responses entered and displayed at the user's terminal. The forms of the interaction can be controlled by the user, with the

display level set according to his familiarity with the system. A request for a "verbose" conversation produces English-like messages from TECCNET. Conversations in "terse" mode, on the other hand, display only short phrases acting as prompts to the user. A user may switch back and forth between conversational modes at any time during his TECCNET session; a comparison between the two forms for the same request is presented in Table 1.

Table 1 Sample TECCNET Interaction: How to Insert Commands

```
****USER::
help comment

++++TECCNET:
    The comment command is your way of recording comments throughout your
 TECCNET session.  Any information may be placed in a comment field (i.e.,
 difficulties with the system, documentation of the experiment being
 performed, etc).  If you type      comment        ,TECCNET will
 respond with a special comment prompt (++comment: ?).  The cursor will
 remain after the ? and you may begin your comment.  At the end of a line
 hit the carriage return to receive your next comment prompt and enter
 the next line of your message.  The format is free form, anything may
 be included in your text except the error correcting characters (@ and #).
 When you are through, merely hit carriage return immediately after
 receiving the prompt.

****USER::
terse

++++TECCNET:
 Your request for terse output is acknowledged.

****USER::
help comment

++++TECCNET:
    If you type     comment    TECCNET will respond with a special comment
 prompt (++comment:?).  You may begin typing.
```

The Conversational Interface is basically command driven; a feature which gives a user considerable flexibility in his use of the system. This is a "user active mode", of the style generally preferred by the designers of interactive

systems. This preference is due in part to the speed and brevity which can be achieved in the interaction. More fundamentally however, it reflects the fact that experienced users can learn to use the system efficiently without incurring interactive overhead. If this active mode is used throughout, however, the burden of choice is placed on the user. Therefore, at selected points within the TECCNET system, when complex descriptions or consistent specifications must be solicited from the user, a more restrictive question and answer (or user passive) format is employed.

Whenever the user message (****USER::) appears, TECCNET is awaiting input from the user. An interpreter monitors the user's entry to distinguish the following: 1) signals for movement within TECCNET (motion commands), 2) requests for information (help commands) and 3) specific data entries (response to system prompts).

Motion commands allow the transfer between the basic user activities. For example, the command "model" places the user in a position to define his modeling environment; while the command "scenario" allows him to specify his test case. Motion commands need not be executed in any particular order nor given within the same TECCNET session. Although ultimately the execution of a given experiment (activated by the "run" command) requires that all modeling and scenario specifications have been completed, no constraints are placed on the user as to when these specifications are made. The final experiment draws on pre-stored files; files selected from those created during model and

scenario building activities that have been completed prior to execution.

Motion commands achieve more for the user than a shift from one function to another; they generate interrupts for the TECCNET system. As a result, motion commands can be issued to escape from even the most detailed level of a tree-structured question and answer sequence. The effect of this interrupt is to cancel an activity in progress, such as file creation, data entry, or execution, before returning control to the user and repositioning him within TECCNET. When all activities associated with cancellation have been completed, the user is prompted to proceed with the appearance of a user cue. The BREAK or ATTN key generates a pseudo motion command. It creates a system interrupt with all of the cancellation features indicated above. However, the user is moved to a temporary transfer point only, at which time he will be expected to determine what to do next.

A user is supported during his session by a number of help commands which provide on-line documentation and clarification, and which may be issued at any time. Unlike motion commands, help commands have no positional properties, and are not true interrupts. When a help command is entered, the information requested is displayed and the user can continue his session from his current position. If a question had been asked, the question is repeated; any number of help commands can be issued. An example of such a command, indicating how to correct errors in the command line, appears in Table 2.

Help commands can also be used as an on-line training

Table 2 Help Commands Supporting Error Correction

```
****USER::
help error

++++TECCNET:
     If you detect an error BEFORE you strike the carriage return key
 you may correct it in one of the following ways:
   1-Type the AT SIGN (@) to cancel the command line up to that point.
 Immediately retype the correct line.  Thus,if you type
          shop@stop
 you will get the 'stop' command correctly.
   2-n NUMBER SIGNS (#) will cancel the last n characters you typed.
 Thus, either of the two lines
      sh#top
      shop###top
 will enter the 'stop' command.  Combinations also work;e.g.
      shop###toa@show@stop

     If you wish to correct an error AFTER you have entered your
 command or to interrupt TECCNET, strike the BREAK or ATTN key
 and wait for the user cue.
```

mechanism for the user. An underlying structure for command use is presumed in such an exercise. A next logical command in the sequence is suggested at the end of each response; however, this sequence is in no way compulsory for the user. The help commands, as indicated previously, are part of the active command portion of the Conversational Interface, and the parser and command interpreter support the system in an order-independent way. A partial sequence of commands and responses, used as an introduction to the system, is depicted in Table 3; while a sample TECCNET session appears in Appendix A.

The question and answer format represents the third type of user interaction. The order of responses here is critical; a request to the user for a "yes" or "no" answer, for example, blocks the progress to the next branch of the logic tree until an acceptable response is given. In this case, the

progress of the user is controlled by the system, hence the designation "user passive". To a certain extent, as indicated above, a blend of active and passive user modes is achieved even in this interactive form. Despite the fact that specific answers are required, help commands will be honored at any time by the system, and the user may issue motion commands to terminate the tree sequence and regain control.

The parser that supports the question and answer interaction is, of necessity, more complex than the order independent interpreter. The order-independent interpreter poses only two basic questions internally before proceeding or printing an error message to the user: 1)is the input from the user an interrupt request (motion or pseudo motion command)?, and 2) if not, is it a recognizable help command? A negative answer to both questions leads to the conclusion within the parser that either 1) the user is confused, in which case he needs a reminder of what is expected, or 2) he has made an error in typing. A typical sequence exemplifying the reponses from the parser to an incorrect command is depicted in Table 4.

The question and answer parser applies the same two tests as the order-independent interpreter. However, in this case a negative answer to both internal questions leads to the hypothesis that the user input is data and must be processed acording to the rules for the specific question asked. In processing the data, four basic types of entries are recognized; character data which may include numbers or symbols, integer data, floating point data, and binary data (i.e,

Table 3 Introduction to TECCNET: A Tutorial

```
++++TECCNET:

    Welcome to the Information Flow Network Testbed (TECCNET).
 For information on how to use the system, type:
          help
 followed by a carriage return.  Otherwise, on receiving the
 user cue (****USER::) you may type any TECCNET command.

****USER::
help

++++TECCNET:
     TECCNET is an interactive testbed which is intended to support
 the analysis of a number of information flow related issues.  It is
 structured to provide user support in three areas:

          1) the specification of his modeling environment, selection
             of local node models, message protocols and flow control
             strategies.  (program segment:  'model')

          2) the generation of simulation input and scenarios that define
             the network / traffic conditions. (program segment: 'scenario')

          3) the on-line execution of the simulation experiment (program
             segment:      'run').

    For information on how to interact with the TECCNET system, the
 inexpert user should type:        use

****USER::
use

++++TECCNET:
    You converse with TECCNET by entering commands and responses from your
 terminal.  When the user cue (****USER::) appears, it is the indication
 that you may begin typing.  To signal the computer that you have completed
 your entry you MUST strike the carriage return (CR) key.  The computer will
 not respond until you do.
     The commands permit you to move freely within the TECCNET system,
 to exercise the various programs, or to receive explanations and
 assistance in program selection and data preparation.  Occasionally,
 a specific response is required.  In these instances, you will be
 prompted from the terminal before the user cue is given.
     Depressing the BREAK or ATTN key at any time will interrupt TECCNET
 and return you to a point where you may again enter commands.
   The inexpert user should now type:
                    commands

****USER::
commands

++++TECCNET:
    The basic commands for controlling and interacting with the TECCNET
 system are organized into three groups:

          1) movement within the system:
             model: to change modeling environment
             scenario: to create scenario data
             run:   to execute the experiment
             stop:  to leave the system
          2) on-line documentation:
             help:  brief description of TECCNET
             use:   outline of interaction modes
             help model, help scenario, help run: descriptions of
                    the TECCNET submodels
             use model,use scenario, use run: instructions for their use
             help network, help traffic: description of input parameters
          3) user support:
             help error: how to correct errors in typing
             help comment: how to leave comments about the system
             comment: comment mechanism
             terse: request for brief interactive reporting
             verbose: request for complete interactive reporting
```

Table 4 An Incorrect Command

```
****USER::
}run

++++TECCNET:

 Your input cannot be interpreted as entered.
 If you need a list of active commands, type:      commands
 If you wish to leave the system; type:           stop
 Otherwise, retype your command and try again.

****USER::
run

++++TECCNET:
 Your request to transfer to the 'run' portion of
TECCNET is being processed.
```

yes/no,true/false, etc.). If the input satisfies the specifications of the specific question asked, the appropriate action is taken and the question/answer sequence continues. If not, the data format is reprinted, along with the error message indicated in Table 4. An example of a simple question and answer sequence, drawn from the simulation portion of TECCNET is illustrated in Table 5.

2.3 NETWORK MODELING USING TECCNET

Beneath the Conversational Interface are the TECCNET modules that define the modeling system. Here, as in the creation of the interactive software, two major concerns dominated the design process: 1) the desire to understand the needs of the users (both current and future), and 2) the intent to structure a system that could be expanded as these needs change. Unlike the design of the user interface, in which plans

Table 5 Question and Answer

```
++++TECCNET:
    Please enter the name of the file in which your network
 data has been stored.  If no file exists or you wish to create
 a new file,  type:        new
 You will be prompted by the scenario builder for the data which defines
 your network.
 If you need clarification, type:       !help network

****USER::
net2

++++TECCNET:
    Please enter the name of the file in which your traffic
 data has been stored. If no file exists or you wish to create
 a new file,  type:           new
 You will be prompted by the scenario builder for the data which defines
 your traffic conditions.
 If you need clarification, type:       !help traffic

****USER::
traf2

++++TECCNET:
 Network and Traffic input complete. Would you like it displayed?
                     Answer yes or no

****USER::
no

++++TECCNET:
 Please indicate the desired number of iterations
                     Enter an integer >0

****USER::
5
```

for expansion can be met through simple additions to the message set, the remainder of the TECCNET system required that explicit provisions for expansion be made as part of the initial implementation. Capabilities intended to support future development were incorporated using programming "stubs", or data structures and empty procedures. This approach was taken in order to insure a complete functional implementation according to the design illustrated in Figure 1. In the sections that follow, each of the user functions indicated in this diagram are described, their current form is presented, and expansion provisions are indicated.

2.3.1 Specifying the Model

In order to represent the information flow network, the user must first initialize the simulation. This entails, as indicated previously, the specification of the model of the processing elements at the node, the messages and protocols defining the information flow and the algorithms for managing the network. In general, these three specifications are tightly coupled, bound together by the need for consistency in the modeling assumptions. Even within these contraints, however, some flexibility exists in constructing a modeling environment.

Before considering potential modifications, a clear appreciation of any built-in assumptions must be developed. Default specifications for the processing model are currently in place within TECCNET. These defaults permit one type of analysis of an information flow network by viewing it at its lowest level; that of the nodes and links comprising a store and forward data communication network. In this case, therefore, the queueing theoretic model of the processing elements (particularly the node) used is extremely simple. It is based on the following key assumptions: Processing of data packets takes "zero" time compared to packet queueing and transmission delays. The link buffers at the nodes are not modeled explicitly; their capacities are reflected in an effective link capacity that is a fraction of the physical limit of the line. Moreover, the transmission and receipt of packets are assumed to be perfect processes.

Characteristics of the message traffic (exclusive of

volume) also have been specified as defaults. For simplicity, data packets are assumed to have the same average length, and only one conversation may be active between a pair of nodes at any given time. Knowledge of the structure and content of these data packets is not required in the initial analysis. Control messages, on the other hand, require explicit treatment of both structure and content. The header information included is typical of that contained in the control packets of actual networks (e.g., source, destination, transmit time, packet class, etc.). The message content of the control messages is algorithm specific.

The choice of message protocols completes the message specification. A first-in-first-out (FIFO) service discipline is assumed for the treatment of data packets. Both control packets and acknowledgements, on the other hand, are assumed to have high priority in the system; implying that either they are sent out immediately or "piggy-backed" on data packets waiting at the head of the queue.

The content of the control messages is used to drive the TECCNET algorithms. The initial network management algorithm, an outgrowth of an original routing scheme developed by Gallager [GAL77], was first outlined in its current form by Golestaani in 1980 [GOL80]. This procedure combines both routing and flow control mechanisms in a single distributed algorithm. From the point of view of the future developments of the information intermediary, as outlined in Section I, the following features of the algorithm motivated its selection as the initial

network management tool: 1) the type of marginal delay information passed between local nodes, and 2) the structure of priority functions that represent the cost of rejecting flow between individual node pairs. A description of the flow control and routing algorithm, and its implementation, will be presented in Section III.

Even for this simple set of specifications, a number of modifications can be made that will change the modeling environment substantially. The most obvious is a alteration to the flow control algorithm, consistent with existing model and message specifications. These changes would be cataloged as additional entries in the flow control algorithm library which will permit them to be retrieved readily by the user for inclusion within the simulation. (The initial algorithm is currently the sole entry in this library and is retrieved automatically). Less obvious modifications, but still ones leading to a different set of model specifications, might include, for example: 1) the incorporation of a non-zero processing delay for control (as opposed to data) packets, along with an expanded node processing model, or 2) a change in the priority associated with the movement of control information.

The approach to simulation building outlined above is extremely appealing, given the plan for the use of TECCNET to support C3 research. However, a note of caution regarding the use of building blocks for specifying the modeling environment is appropriate. As the scope of the processing models broadens, the number of options to be considered will grow dramatically.

Extreme care will be required on the part of the modelers introducing enchancements to TECCNET to insure that the selection criteria for initializing the simulation are straightforward, and that at all times the user can obtain a clear picture of the modeling assumptions being used.

2.3.2 Scenario and Input Generation

From the specification of the model, one moves to the creation of the scenario, supported by the second of the major modules within TECCNET. In general, one assumes that the sequence of the TECCNET functions is a nested one in the following sense: the user wishes to specify a model, then test it against several scenarios, each of which is is used to drive to a number of experiments. However, one could readily imagine alternative organizations; ones, for example, calling for the evaluation of various versions of a model against the same scenario.

Therefore, it was recognized at the outset that any sequencing assumption in organizing an experiment was a potentially limiting one. Care was taken, first in the design of the Conversational Interface and later in the development of the Model and Scenario Generators, to support the desired order-independent processing. The most significant effects of this need for flexible ordering were seen in the design of the support mechanisms for scenario specification, particularly in the areas of the cataloging and storage of the input data. In the section that follows, these mechanisms will be described in

parallel with the presentation of the scenario building activity.

Within the scenario specification section of the system, the following types of input are required: 1) values of model parameters that define network operating characteristics, and 2) a description of network structure, traffic, and environmental conditions. As the modeling of the information flow network elements becomes more sophisticated, additional classes of inputs may be added. However, in adding to the list of variables under control of the Scenario Generator, scenario inputs (describing the condition of the information flow network) will continue to be distinguished from those that define the experiment (such as number of iterations, convergence criteria, cost function parameters, type of statistics to be collected, etc.). This distinction is best appreciated by the user who is attempting to combine "canned" scenarios and model specifications for use in multiple experiments.

Inputs are solicited from the user and organized into permanent files. The scenario building process may occur in small segments, at different TECCNET sessions until a complete scenario has been obtained and stored in the system. The major focus of the user support provided during this process is placed on the preparation of the description of the network topology; often the most tedious aspect of data entry for network modeling.

The convention was adopted that all network elements, (nodes and links) are assigned unique, order-independent names by the user. This eliminates the need to define an element in terms of a number indicating location in a specified network; a

position that changes with each addition or deletion. Data entry, therefore, is equivalent to defining an unordered data base, containing a superset of the elements (in this case a list of communication links and input requirements) the user wishes to use in creating a number of scenarios, and from which subsets may be extracted without respecification. A number of different network data files, required to depict the network topology, may be created from the same data base. Each data base, as well as each data file, contains descriptive information characterizing the type of network on file (solicited in free text from the user at his terminal), and data.

A data base is built by entering descriptive information characterizing the network elements from the terminal. Element data, required for the model specification currently in TECCNET, are relatively simple. A link entry in the data base, requires a link name, node names of the origin and destination nodes, and a nominal effective capacity. The units of this capacity are not supplied, the only requirement is that capacity and flow data be consistently specified (whether in bits/second or kbits/second) throughout the experiment. These data are input according to a simple free format that is described to the user at his terminal. A typical input line, with the entries separated by commas, illustrates the link and node names and capacity information (i.e., L21,NSAM,N24,5.,). Since the format is simple, with no special data management characters included, data base files may also be generated by a computer program, of interest to the user who may wish to

Any network file, no matter how modest, may, if desired, function as a data base for network file building activities. The interactive and storage formats are the same for both the general data base and specific network data files. In order to abstract a subnetwork from any network file, a user merely indicates a set of links to be extracted. He may then augment this list with link input from the terminal, as illustrated above, to define a new variation of his scenario. An example of an interaction of this type is decicted in Table 7.

Similar support is provided for the generation of traffic files. In this simple modeling exercise, node names defining conversation origin and destination must be supplied along with the average desired input rate (in units consistent with the link capacities). Since the algorithm under study in initial version of TECCNET uses expected values only, no further statistical description of the input characteristics is required. Unit multipliers are generated as placeholders to describe the potential traffic variations and are attatched as part of the scenario.

The environment is also specified in terms of unit multipliers. In this initial version, capacities are assumed to remain constant (no degradation due to environmental effects), and nodes and links are assumed to be operational for the duration of the experiment. Minimal provisions for the generation of alternative time functions, and probabilistic descriptions of the input are included; these will be expanded as the need arises.

Table 7 A Network Data File

```
++++TECCNET:
    Please enter the name of the file in which your network
 data has been stored.  If no file exists or you wish to create
 a new file,  type:        new
 You will be prompted by the scenario builder for the data which defines
 your network.
 If you need clarification, type:       !help network

****USER::
net2

++++TECCNET:
    This file already exists, do you wish to treat it as a data base?
                    Answer yes or no

****USER:
yes

++++TECCNET:
     Please indicate the name to be assigned to your new network file.

****USER:
net2a

++++TECCNET:
     Indicate a list of link elements to be extracted from this data base
by typing the link names, separated by commas.  Terminate your list
by typing:            end (CR)
          Example:    11,12,142,L60,    (CR)
                      end

****USER:
11,12,13,14,
end

++++TECCNET:
   New elements are added using the basic input format.
Network data are entered one link at a time, with a carriage return (CR)
after each link.  Four data items are required; a link name, a node
name for the origin node, node name of the destination node (each < 6
characters) and the effective capacity of the link.  Data items are
separated by commas, as entered in the following example:
          L21, Nsam, N24, 5.,
  Input is terminated by typing:  end      followed by a carraige return (CR).

****USER:
15,n2,n3,15.,
16,n3,n2,20.,
end
```

2.3.3 Event Generation

Once a model and scenario of interest have been established, the execution phase of the experiment can be initiated. Processing order is important within this module, and the question and answer format is the predominant one in completing the specification of the experiment. Discrete event simulation techniques form the basis of the execution software, which permits the integration of many procedure-driven models and the represention of asynchronous operation of the elements of a distributed system.

Three types of events are modeled, designated for purposes of discussion "spontaneous", "responsive", and "external". Both spontaneous and responsive events refer to conditions occurring within the nodes of the information flow network. External events, on the other hand, refer to situations arising outside the network which are seen by the system only in so far as these conditions have a measurable effect on the real-time capabilities of the system elements. It should be noted that these external events, although important conceptually and therefore supported by the event generator, are not required by the simple models used in the testing of TECCNET, and are therefore considered part of the future system.

Spontaneous events simulate actions that are based solely on internal logic operating within the elements themselves. No direct outside stimulous is required; thus, these internal events correspond to the decoupled actions of a

cooperating member of a distributed system. These events may take many forms depending on the modeling environment that has been specified. The most straightforward of the spontaneous events is a scheduling event, such as the initiation of a particular process at a node. An internal clock or algorithm may be used to determine the activation time for the "next planned" execution of an event, and may in turn, be used to schedule communication with other nodes. A slightly more elaborate form of scheduling event is a conditioned one, in which some quantity, observable at the node, is monitored until a threshold is reached, at which time the event is scheduled. Only the first type of spontaneous is required to support the models currently implemented in TECCNET. The primary spontaneous event in this case is the command from each node to initiate a routing/flow control update cycle which will govern the flow of traffic to it from other nodes in the network. These events are scheduled by the nodes using independent internal clocks, which are not synchronized. (Synchronization of the internal clocks may be requested by the user to test specific hypothesis, however, it is not a general requirement for most of the distributed algorithms to be examined in the C3 context). The execution of each of the spontaneous events results, in the case of the current algorithm, in the generation of a set of "transmit" events simulating the initial broadcast of the fact that an update cycle has begun. A second spontaneous event, the generation of a special flow probe packet, described in Section 3.2, is triggered by the same type of scheduling mechanism.

All other events currently implemented are responsive ones, in that an external stimulous in the form of a message arriving at a node is required. A node receiving a message from another node must respond; the nature of that response depends entirely on the content of the message, his message history, and the algorithm describing his processing. A node may be waiting, for example, for the arrival of the first message of a given type, or the last message from a set of nodes of interest, before taking some action (i.e., running a process, updating a parameter, or creating a message for transmission). These responsive events currently included in TECCNET will be presented in more detail in Section III where the procedures describing the distributed routing/flow control algorithm are described.

Before proceeding to the discussion of this particular algorithm, a few remarks may be appropriate regarding the support provided within the event generator for model development and expansion. The first of these is the list management software which supports the addition and deletion of events from the event list and drives the simulation according to the next event in time. This software is generic, new event types can be defined by a developer interacting with TECCNET, and the corresponding processing options added to the event list routine. Additional algorithm development tools are also supplied, based on groups of logical statements often needed by the algorithm implementor. These describe events beyond simple message transmision and reception; examples of these events are given as follows for both event generation and event monitoring: 1) transmit message

to adjacent nodes, 2) transmit message to upstream (1) nodes only, 3) indicate when control packets have been received from all downstream nodes, or 4) indicate when the first message of a given type has arrived.

The event generator as currently implemented is only partially interactive. The user is on-line while the simulation is running so that he may view the results and possibly decide to abort the experiment. However, the type of interaction possible during simulation is limited to output control, honoring requests to suppress or display output at various levels. For the future, a true interactive node is envisioned in which the user will be changing the inputs in real time as though he were a network customer. This feature appears in the current system as a software stub.

(1) "upstream" is defined as the set of nodes from which a node receives packets for a given destination, "downstream" are those nodes to which those packets are sent.

III. THE NETWORK FLOW CONTROL ALGORITHM

3.1 SPECIFICATION OF THE ALGORITHM

In the preceding section the TECCNET modeling system was presented, with emphasis on its structure and how that structure has been implemented to support C3 system research. In the process, the initial modeling environment with its built-in assumptions regarding the type of experiment to be performed was described. This description presumed that a low-level view of the information flow network would be taken and that the insights gained from the network experiments would be used to develop local models of the network suitable for inclusion as part of the Information Intermediary. It was clear that within this framework, a broad class of techniques for managing the flow of information through the network could be studied. Of particular interest were those which could be implemented in a distributed manner and for which the control actions and decisions taken by the individual nodes required limited local information.

The initial algorithm included in TECCNET is one proposed by Golestaani in his Ph.D. thesis [GOL80]. Using this approach, flow control and routing are treated together, leading to a flow control algorithm with two components: a quasi-static portion and a dynamic portion. The quasi-static formulation is an outgrowth of an earlier work by Gallager [GAL77], a distributed algorithm for determining optimal routing in a communication network. In Golestaani's extension, the

combination of optimal routing and flow control expresses the following conflicting network management objectives: one attempts to reduce congestion in the network while minimizing the amount of offered traffic that is rejected by that network. In the remainder of Section 3.1, this approach is described and, at selected points, the specifics of the TECCNET implementation are indicated.

A convex optimization problem is formulated in which short-term average information on network utilization is used to allocate both maximum data rates for each user session (viewed as source/destination pairs) and the optimum routes through the network for information flowing within it. The description of the formulation requires the following definitions: The network structure is indicated by 1) a set of nodes {N} in which each of the elements is indexed by an integer (i) and 2) a set of links {L} connecting these nodes, each of which is designated by a pair of indices, km (denoting a source node (k) and a destination node (m)). Each link is completely characterized, for purposes of the simple model required by the algorithm and supported by TECCNET, in terms of an effective capacity, C_{km} , (1) and the flow carried, f_{km}. A set of commodities is specified to represent active conversations between individual node pairs. Each of these commodities (designated by indices ij to denote the source and destination nodes) is described by a desired input rate, r^{d}_{ij} ,

(1) Effective capacity represents the maximum link flow that can be handled realistically by the link, given the number of buffers at the nodes (at both the transmitting and receiving ends). Typically this flow level will be slightly less than the physical capacity of the line.

which the flow control scheme attempts to satisfy. The rate allocated to each session is indicated by r_{ij}, and may take on values up to and including the desired.

The allowable transmission rate for each session is determined through consideration of user priorities, fairness, and the expected level of congestion throughout the network. A balance among these considerations is achieved through the appropriate selection of cost functions used in the optimization. The interpretation to be placed on each of the cost functions and the mathematical motivation for the requirement that they be twice differentiable and convex are described elsewhere [GAL80] [GOL80]. In this discussion it is sufficient to indicate the following: Two costs are included; the first, that of congestion, is associated with each individual link in the network. This cost of congestion is represented by an increasing function of the volume of flow on that link and, in this instance, has been chosen to be:

$$g(f_{km}) = \frac{f_{km}}{C_{km}-f_{km}} \tag{3.1}$$

The second cost, that of user dissatisfaction, is associated with each user session originating between node pairs. It is represented by a decreasing function of the allocated data rates, $e(r_{ij})$, and may take many forms.

The individual link and session costs are combined to form an overall network cost given by:

$$J = \sum_{k,m} g(f_{km}) + \sum_{i,j} e(r_{ij}) \qquad (3.2)$$

The allowable session rates and desired routes through the network are chosen to minimize this overall cost. The conditions for optimality [GAL80] depend only on incremental link costs (indicated in equation 3.3) and a priority function, $-e'(r_{ij})$, associated with each conversation.

$$g'(f_{km}) = \frac{C_{km}}{(C_{km}-f_{km})^2} \qquad (3.3)$$

A desirable form for the priority function (and the one implemented in TECCNET) is given by:

$$-e'(r_{ij}) = s\left(\frac{a_{ij}}{r_{ij}}\right)^{b_{ij}} \qquad (3.4)$$

The scaling factor, s, is a parameter which is used by the TECCNET simulation to represent a global balance between the concerns for network congestion and rejected flow. The parameter a_{ij} is a measure of a typical rate for session (ij), while b_{ij} is a measure of its importance. [GAL80] [BER81]

It is convenient in constructing the algorithm to add a fictitious link to the network for each active commodity. Structurally, these links connect origin/destination pairs (referred to as ij pairs) and "carry" rejected flow only (up to the desired input rate, r_{ij}^d , indicated above). The marginal cost associated with flow on this link is then equal to the value of the priority function given in equation 3.4.

Expressed in this way, the minimization of the cost function (3.2) is a quasi-static routing problem for which a variety of distributed algorithms have been developed. [GAL77] [BER81] [SEG79]. Readers are referred to [GAL77] and [GOL80] for a formal description of the particular algorithm selected for implementation. A brief presentation is included as part of this discussion for completeness.

Two phases of the algorithm are required to achieve a complete update of the routing variables. The first phase is a "protocol" phase, in which the nodes pass selected network status information through the network according to established procedures. The second is a "commit" phase, in which nodes implement the desired changes in routing and input control and inform the other nodes in the network.

Within TECCNET, a "one-at-a-time" update policy is used. That is, each destination node j initiates its own update cycle-- one which governs explicitly only the flow associated with conversations directed to it-- independently from the other nodes. (1) An update cycle begins with the following steps:

Protocol Phase

1) Node j broadcasts an initiation message containing time information to all adjacent nodes (nodes connected directly to it). This broadcast is a spontaneous event as described in

(1) While the actions are independent, we are reminded that the status information on which they are based is not. It is the total flow carried on the finite capacity links that is reflected in the marginal cost information transmitted through the network.

Section 2.3, triggered assording to an internal clock at each node. The interval between updates is controlled by the user.

2) Each node k knows the identities of nodes to which it currently routes flow destined for j (designated the set of all downstream nodes {M}). After receiving the FIRST protocol message initiated by node j, the node begins monitoring its protocol traffic and waits until it has received a protocol message from all downstream nodes. These protocol messages, received from each node in {M}, contain the following information:

- the value of the initiation time from node j
- the value of the marginal cost, (dJ/drmj), associated with flow between node m in {M} and the destination j.
- a flag indicating whether, given the current status, node m should remain on the path from node k to node j. (This mechanism, a "blocked flag", is a required to prevent the formation of loops in the path logic between iterations [GAL77]).
- estimates of the average delay per packet, T_{mj}, for packets traveling between node m and j. This estimate is required in the calculations of windows which are used by the dynamic portion of the flow control scheme. The specifics are given later.

3) Each node k then updates the marginal cost from itself to node j according to the following:

$$\frac{\partial J}{\partial r_{kj}} = \sum_{m} \phi_{km}(j) \left[g'(f_{km}) + \frac{\partial J}{\partial r_{mj}} \right] \tag{3.5}$$

where $\phi_{km}(j)$ represents the fraction of total flow at node k (including all desired inputs) routed to node j along link km,

and g' is defined according to equations 3.3 and 3.4. The average delay, T_{kj}, is updated in a similar manner, combining locally generated estimates of the delay between k and nodes in {M} with estimates received, T_{mj}. A protocol message, containing these estimates is sent to all neighbors NOT in {M}.

4) When node k has received a protocol packet from all adjacent nodes, it can infer that the protocol phase upstream from it is complete. Node k then sends protocol information, which acts as a confirmation of this fact, to all nodes in {M}.

5) Finally, when node j receives protocol confirmation from all upstream neighbors, all nodes have updated their estimates of network conditions in response to the update request. Node j need not wait for communication from the remaining adjacent nodes (those from whom it receives no flow) before initiating the commit phase.

Commit phase

1) Node j broadcasts the update signal to all adjacent nodes.

2) Node k waits only for first receipt of an update signal, at which point it:

- updates routing variables for all outgoing links, including the fictitious link. (In the current implementation, this update is performed in the simplest manner [GAL77]. More elaborate procedures can be implemented as indicated in [BER81]).
- updates window size as defined below
- applies new values and progates flow

The information required to support each of these calculations is translated into an appropriate control message. These messages are used to generate both transmit and receive events in the simulation. These events are associated with specific nodes, according to the logic outlined above, and are scheduled in time using the expected processing and transmission delays for individual links.

The reference, in the preceding description, to window updates and the passing of delay information through the network reflects the requirements of the dynamic part of the flow control algorithm. Dynamic flow control has the function of admitting or rejecting individual packets into the network so that the allowable rates, as determined by the optimization algorithm, can be met and fluctuations in arrivals and buffer occupancies can be smoothed out. Various suggestions for the computation of the window sizes, the mechanism for achieving this control on individual ij sessions, have been made [GAL80] [GOL80]. In developing the algorithm, one begins with a relationship for determining the window size, w_{ij}, given by:

$$w_{ij} = \frac{r_{ij}}{\Gamma}\left[T^{+}_{ij} + \theta_{ji}\right] \qquad (3.6)$$

where Γ is the average packet length, θ_{ji} is the time required for an acknowledgement from j to i. T^{+}_{ij} is a revised estimate of the average per packet delay between i and j; the superscript + indicates that it is based on delay observed <u>after</u> the new control variables have been instituted, and new flows have

resulted. If this computation were attempted directly, passing all required flow and routing information in the same distributed manner given above, three times the control information (effectively three complete cycles) would be required to achive one complete update [GOL80], clearly not a desirable result if one is trying to conserve network resources. An approximation is made in this implementation which permits window sizes to be updated as the new allocations of the routing variables are determined. Values of T_{ij} are computed during the protocol phase of the algorithm, and continue to be modified between protocol phases using timing information contained in the network acknowledgements. When the new value of r_{ij} is determined, the window is computed using the following:

$$w_{ij} = \frac{r_{ij}}{\Gamma} \left[T_{ij} + \theta_{ji} \right] \tag{3.7}$$

The use of T_{ij} as an approximation for T^{+}_{ij} has been justified on the basis that the update policy has been implemented asynchronously in TECCNET. As a result, the effect of routing changes, applied to traffic other than that destined for j, is likely to have taken effect. These changes are also likely to have been reflected in the estimate of T_{ij}.

Implementation considerations are outlined in the section that follows. Experience with the implementation in which this approximation is used is described in Section 3.3.

3.2 IMPLEMENTATION ISSUES

The blend of dynamic and quasi-static techniques into a single scheme poses an implementation challenge. As indicated above, the windowing procedure is intended to govern the entry of individual packets into the network. As a result, this mechanism is most easily simulated packet-by-packet, using discrete-event simulation techniques. The steps which comprise a single protocol/commit cycle of the quasi-static algorithm are also driven by the movement of individual packets (albeit control instead of data) in the network. Therefore, even if the data packets are not considered explicitly in generating the flow estimates required during the optimization, a sequence of discrete events still represents the most straightforward translation of the quasi-static portion of the algorithm into a form for evaluation.

There are many well documented techniques for applying discrete event simulation to the analysis of networks, and elaborate languages have been developed to suport their use [ORE80]. Although not designed expressly for the purpose of packet-by-packet network simulation, TECCNET can be used to represent the network at this level of detail by virtue of its discrete event execution structure. However whether one uses TECCNET or some other simulation tool, considerations of cost and manageablilty of the experiment must be addressed.

In general the data and control packet simulations require the explicit treatment of large volumes of traffic, simulating the movement of packets in some detail through the

network. In the case of this particular joint routing and flow control algorithm, an open simulation would require that the data packets be 1) generated for each session according to the appropriate distribution, 2) injected into the network at the source node each time a session window became available, 3) transported through the network along with the other data, control and acknowledgement traffic, 4) queued at intermediate points, and 5) ultimately removed at the destination. For a network of modest size, tens of thousands of individual transmit, receive, or acknowledgement events may be generated during each complete update cycle (protocol/commit phases) for the overall network. Morever, if Monte Carlo analysis is required (due, for example to the event structure, disturbances in the system, or interactions modeled) the execution of enough cycles to give confidence in the experimental results would add considerably to the computational burden.

It may be argued that the full packet simulation yields improved fidelity and modeling flexibility over various forms of analytical models. Depending on such things as the complexity of the delay model implemented, the detail desired in the representation of the procedures at the nodes for responding to specific messages, and the approach required for estimating flow at the nodes, this argument may be more or less valid. However, it is important to recall that one of the significant objectives in developing TECCNET as a research tool is to admit the possibility of including the user as an interactive agent in the experiment. Clearly, therefore, the implementation approach

taken must be such that interactive execution is practical. Thus an alternative to the full packet simulation was sought which would retain the strengths of the discrete event formulation in capturing the interactions between the control variables and the flow in the network.

In considering the quasi-static portion of the algorithm, the modeling approach was drawn from analytical results obtained using similar distributed routing algorithms. One experiment in particular suggested that, under certain conditions, a reasonable comparison could be made between the results of the analytical and discrete-event simulations of the same algorithm, despite the fact that the analytical simulation assumed no correlation between events at the various nodes [CAI80]. With these results in mind, the following technique was used for the test cases described in Section 3.3.

Control packets that drive the algorithm are modeled explicitly, scheduling events according to the estimates of the single link delays incurred and the logic of the protocol/commit sequences indicated above. Control acknowledgements, specified as part of the algorithm, are also modeled explicitly. The bulk of the traffic, however, is represented in an aggregate way only. Specially designed pseudo-packets (designated "flow probes)" are created as an artifice of the simulation. These packets are added to the event stream at the appropriate times to carry information that permits flow estimates to be updated at a node. As these probes move through the network, they reflect the various changes in flows that are the result of modifications in

routing or inputs. These packets function as aggregate data tokens, and in the limit, if the time between packets were random and small, the effect of these probes would be roughly equivalent to the packet-by-packet model. The user determines the frequency of flow probe initiation at a node, however, special care is taken to see that probes follow control packets issued during protocol and commit phases of the algorithm. The benefits derived from this approach are seen in reduced execution time for the simulation; potential losses in accuracy in the results are expected to be tolerable.

Of greater concern was the implementation of the windowing scheme. It is clear that this form of dynamic flow control requires something approximating discrete packet tokens in order to be successful; e.g, the fewer acknowledgements that are received at the source (as a result of increased round trip delay), the fewer packets will be admitted through the windows into the network. The probes, carrying flow information, have been used to provide information by which these fluctuations can be simulated, even though discrete data packets are not represented. Each probe carries a time tag. With the return of each probe acknowledgement, the actual delay incurred, $\hat{t}_{ij}$, associated with a portion, ξ, of the total flow transmitted from node i to node j can be determined. The effective delay associated with injections at the particular node can be updated recursively using the following weighted expression:

$$T_{ij} = h \left[(1-\xi)\ T_{ij} + \xi \hat{t}_{ij}\right] + (1-h)\ T_{ij} \qquad (3.8)$$

where h can take on values between 0 and 1. If the window size, w_{ij}, remains fixed between update cycles, and the value of T_{ij}, computed according to equation 3.8, is used in equation 3.7, an effective injection rate, $\overline{r_{ij}}$, can be determined each time a flow probe is acknowledged. This effective rate is then used to generate flow information transmitted by future flow probe messages.

In the section that follows, these modeling tools are exercised using two different networks and specifications of user sessions.

3.3 EXPERIENCE WITH THE SYSTEM -- AN EXAMPLE

With the completion of the basic TECCNET system, and the implementation of the initial flow control algorithm, any of a number of experiments (whether leading to the ultimate development of the Information Intermediary or directed toward the exploration of the algorithms themselves) can be designed and carried out. In preparation for the use of the system by various researchers with different backgrounds, a number of exercises were conducted to test the system from model definition through execution. Some preliminary results obtained with two of the sample networks are described in the section that follows.

3.3.1 Case 1

The first test was performed using a small (5 node 12 link) network, selected for simplicity to verify the accuracy of the event sequences representing the algorithm. The structure of

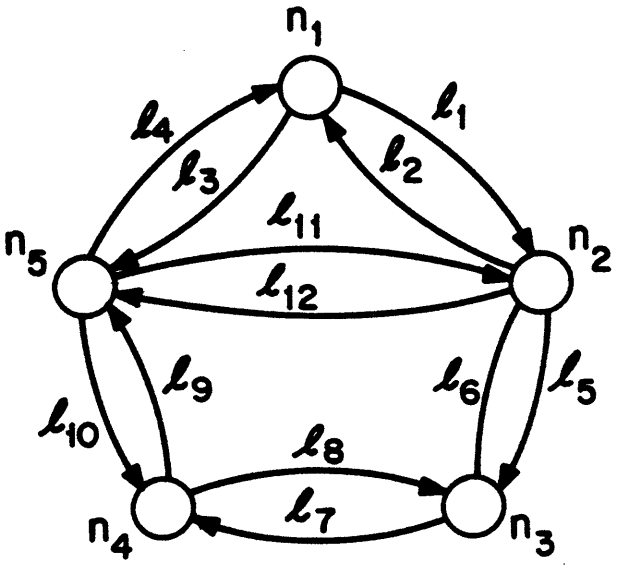


Figure 2 Network Topology: Case 1

this network is depicted in Figure 2, with the nodes and links indicated by labels n1 to n5 and l1 to l12 respectively. The effective capacity for each of the links is the same, set at 100 kbits/second. A single value of .1 kbits for average packet length was used; additional values can be chosen to reflect differences between the various control packets and the average data packet. Control and acknowledgement packets were assumed, as indicated in Section 2.3, to have high priority in the system. Thus they experienced transmission delay only. The desired input rates in kbits/second for each of the 13 individual sessions are

indicated in Table 8. The priority function for these conversations have a_{ij} set equal to the desired rate indicated in Table 8, and b_{ij} set to 2.0.

Table 8 Desired Input Rates: Case 1

Source	Destination	Desired rij
n1	n2	40.000
n1	n3	50.000
n1	n4	40.000
n2	n1	50.000
n2	n4	40.000
n5	n2	50.000
n5	n3	40.000
n3	n1	40.000
n3	n2	50.000
n3	n5	40.000
n4	n1	40.000
n4	n5	40.000
n4	n3	40.000

The algorithm is initialized by setting routing variables such that all traffic is moved along a shortest path to its destination. Thus, for example, traffic from node n4 to node n1 (using the initial routing) would traverse only links 110 and 14. The injection levels are initialized at 25% of the desired input, subject to the constraint that the initial allocation does not cause any link to carry flow of more than 95% of its effective capacity (In the event that this constraint is violated on initialization, the actual starting injections are reduced by a factor of 2 until a feasible initial condition is reached). Once the algorithm is running, this constraint no longer applies.

The initial flows on the links for this test case are depicted in Table 9. With this simple network of uniform

Table 9 Initial Flows on the Links

Link	Flow	Link	Flow	Link	Flow
11	22.500	12	22.500	13	10.000
14	10.000	15	12.500	16	32.500
18	20.000	19	20.000	110	30.000
111	12.500	112	20.000		

capacity links and similar conversations, the initial shortest path routing for each conversation is not far from optimal. Minor routing adjustments during the optimization process reflected the fact that alternative attractive paths existed. Only small modifications were required to equalize the costs due to differences in individual conversation requirements. Thus, although there was some interaction between the routing and flow control variables, most of the action, during the update cycles simulated, was concerned with the admission of additional flow to the network. Within the first three cycles, most of these changes had taken place, stabilizing within eight cycles of the algorithm. The input rates allocated to conversations at the end of eight cycles are presented in Table 10; to be compared with those desired (Table 8). The flows on the links at this point (to be compared with those in Table 9) are given in Table 11.

It should be noted that this represents an extremely low network utilization, approximately 30.4%. This is due to the fact that a large faction of the desired flow was rejected

Table 10 Allocated Input Rates after Eight Cycles

Source	Destination	Ideal rij
n1	n2	34.925
n1	n3	11.445
n1	n4	9.830
n2	n1	35.390
n2	n4	10.872
n5	n2	36.407
n5	n3	11.680
n3	n1	9.465
n3	n2	32.273
n3	n5	8.566
n4	n1	11.153
n4	n5	34.562
n4	n3	36.638

Table 11 Flows after Eight Cycles

Link	Flow	Link	Flow	Link	Flow
11	46.359	12	45.280	13	9.830
14	12.768	15	24.252	16	52.420
17	6.885	18	41.719	19	47.231
110	19.577	111	43.038	112	14.948

(approximately 50%). This reflects the relative balance between the priority functions (according to which rejected flow is penalized) and the cost of network congestion. In this case, the penalty for rejection is relatively low, because the scaling factor, s, for the priority function indicted in equation 3.4 was set at .1.

The scaling factor was then raised to .5 and .75 in two successive runs. The percent of desired flow admitted to the

network increased from 50% to 56% and 57% respectively; while the network utilization rose from 30.4% to 36% and 37% for the two variations. This was achieved with little change in delay, a fact which is not surprising when one compares link flows for these variations (Tables 12a and 12b) with those depicted in Table 11.

Table 12a Link Flows After Eight Cycles, s=.5

Link	Flow	Link	Flow	Link	Flow
11	50.693	12	45.580	13	16.141
14	22.186	15	36.977	16	58.472
17	11.582	18	45.207	19	52.425
110	33.200	111	44.690	112	26.619

Table 12b Link Flows After Eight Cycles, s=.75

Link	Flow	Link	Flow	Link	Flow
11	50.644	12	45.888	13	18.363
14	21.810	15	39.609	16	55.477
17	14.853	18	42.592	19	54.609
110	35.123	111	46.747	112	25.923

3.3.2 Case 2

A second test problem was constructed using the same assignment technique for creating the conversation priority

functions. A different topology was specified, with non-uniform link capacities. This network, with its 8 nodes and 22 links, is

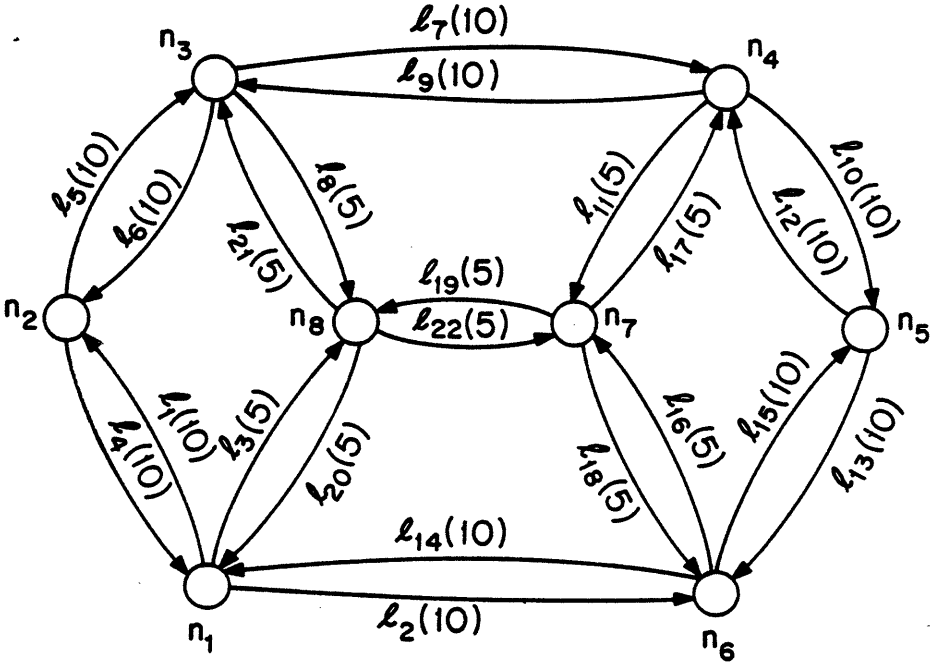


Figure 3 Network Topology and Link Capacities: Case 2

depicted in Figure 3. (1) The desired input rates for each conversation appear in Table 13, while the initial link flows are depicted in Table 14.

The same experiment was run as described above, with s taking values from .1 to .75. Similar results were observed in terms of increases in flow to the network with little effect on

(1) Associated with each link name in Figure 3 is a number in parenthesis, (i.e. (5)) which is a measure of the relative magnitude of the link capacities. The actual effective link capacities, in kbits/second, used in the simulation can be determined by multiplying these link numbers by 10.

Table 13 Desired Input Rates: Case 2

Source	Destination	Desired rij
n1	n2	30.000
n1	n6	20.000
n1	n8	40.000
n1	n5	30.000
n2	n6	30.000
n2	n4	30.000
n6	n2	30.000
n6	n7	40.000
n8	n1	40.000
n8	n3	40.000
n8	n7	40.000
n3	n2	30.000
n3	n8	40.000
n3	n4	20.000
n3	n5	30.000
n4	n2	30.000
n4	n5	40.000
n4	n7	40.000
n5	n1	30.000
n5	n4	40.000
n7	n6	40.000
n7	n8	40.000
n7	n4	40.000

Table 14 Initial Flows on the Links (Case 2)

Link	Flow	Link	Flow	Link	Flow
l1	15.000	l2	20.000	l3	10.000
l4	7.500	l5	7.500	l6	15.000
l7	20.000	l8	10.000	l9	7.500
l10	17.500	l11	10.000	l12	10.000
l13	7.500	l14	15.000	l15	7.500
l16	10.000	l17	10.000	l18	10.000
l19	10.000	l20	10.000	l21	10.000
l22	10.000				

delay. Comparison between the network injections allocated under two different s values (depicted in Tables 15a and 15b) yields an interesting result. The lower rates assigned to long distance conversations such as the session n2 to n4 or n2 to n6 are not surprising. But it is interesting to note that it is the

Table 15a Allocated Inputs after Eight Cycles, S=.1

Source	Destination	Ideal rij
n1	n2	36.567
n1	n6	35.262
n1	n8	19.099
n1	n5	6.348
n2	n6	4.795
n2	n4	4.774
n6	n2	5.904
n6	n7	19.096
n8	n1	19.099
n8	n3	19.099
n8	n7	19.058
n3	n2	36.491
n3	n8	19.037
n3	n4	32.050
n3	n5	7.907
n4	n2	6.464
n4	n5	36.028
n4	n7	19.098
n5	n1	4.656
n5	n4	38.202
n7	n6	19.032
n7	n8	19.099
n7	n4	19.099

Figure 15b Allocated Inputs after Eight Cycles, s=.5

Source	Destination	Ideal rij
n1	n2	34.854
n1	n6	31.466
n1	n8	19.098
n1	n5	10.897
n2	n6	6.874
n2	n4	6.841
n6	n2	8.746
n6	n7	19.099
n8	n1	19.098
n8	n3	19.098
n8	n7	18.218
n3	n2	34.819
n3	n8	18.221
n3	n4	29.739
n3	n5	13.968
n4	n2	9.597
n4	n5	34.477
n4	n7	19.099
n5	n1	6.559
n5	n4	38.200
n7	n6	18.205
n7	n8	19.098
n7	n4	19.098

conversations with the lower allocations which show the greatest improvements as s is increased from .1 to .5.

As indicated at the outset, the preceding discussion is not intended as a definitive evaluation of the flow control scheme or the implementation of the algorithm in TECCNET. Rather it is intended to suggest that a preliminary version of the TECCNET system is operational--one which should support the type of experiments necessary 1) to contribute to the development of local models of the network suitable for the Information Intermediary and 2) to represent the interactions in the information flow networks, inherent in C3 systems.

IV. CONCLUSIONS

In the preceding sections, the design, development and testing of a new research tool, created especially to support C3 system research, was presented. From the outset, the major concern that guided this activity was how the TECCNET system could best support the development of the concepts and models comprising the Information Intermediary.

The design issues for the system were indeed complex, since the potential contributions from a variety of ongoing research activities had to be considered. A consistent framework had to be developed for integrating the relevant notions from research efforts addressing the user/system interface, the generation and management of the information, and the control of the underlying communication networks. Moreover, this had to be accomplished in way that would encourage the exploration of the subtle interactions between the system elements, and the various models, algorithms and procedures that characterize information flow problems in C3.

Preliminary experience with the initial version of the TECCNET system has indicated that these objectives are being met. The interactive format and modular structure of the system appear appropriate to the needs of users with different levels of software and system expertise who will be participating in this activity in the future. The modeling tools incorporated in the system provide the capability for representing the asynchronous interactions and complex protocols inherent in the models and

algorithms likely to be explored. In addition, the presence of a default modeling environment will allow the pursuit of several lines of inquiry in parallel, each of which is expected to contribute from a different perspective to the overall development of the Information Intermediary. It is anticipated that extensive use of the TECCNET system will lead as a by-product to modifications and improvements in the system. As these enhancements are made, it is hoped that the scope of the information flow modeling can continue to increase.

The ultimate success of the approach taken can be measured, on the one hand, by how well the interaction between information strategies and the system parameters can be represented and how readily models of the Information Intermediary can be incorporated and tested. However, as the system is augmented, broader criteria should apply to reflect how effectively the use of TECCNET improves our understanding of the interactions between the distributed command and control network elements, the information flow network itself, and the environment within which the systems function.

V. REFERENCES

[BER80]
Bertsekas, D.P., "A Class of Optimal Routing Algorithms for Communication Networks", Proceedings of the Fifth International Conference on Computer Communication (ICCC80), November 1980.

[BER81]
Bertsekas, D.P, "Notes on Optimal Routing and Flow Control for Communication Networks", Laboratory for Information and Decision Systems Technical Report LIDS-R-1169, MIT, Cambridge MA, December 1981.

[CAI80]
Cain, J.B., D.B. Bradley, and G.H. Thaker, "Simulation for a Quasi-Static Routing Algorithm for Packet-Switched Networks", NTC Conference Proceedings, Houston TX, June 1980.

[COR69]
Corbato, F.J., "PL/I as a Tool for System Programming", Datamation 15 6 May 1969, pp. 68-76.

[COR72]
Corbato, F.J., J.H.Saltzer, and C.T.Clingen,"Multics--The First Seven Years", AFIPS Conference Proceedings 40, 1972, SJCC AFIPS Press, Montvale NJ,pp.571-583.

[GAL77]
Gallager, R.G., "A Minimum Delay Routing Algorithm Using Distributed Computation", IEEE Trans. on Communications, Jan. 1977, pp.73-85.

[GAL80]
Gallager, R.G. and S.J. Golestaani, "Flow Control and Routing Algorithms for Data Networks", Proceedings of ICCC'80, Atlanta GA, October 1980.

[GOL80]
Golestaani, S.J. "A Unified Theory of Flow Control and Routing in Data Communication Networks", PhD Thesis LIDS-TH-963, Laboratory for Information and Decision Systems, MIT, Cambridge, MA, Jan. 1980, 92 pages.

[HUA82]
Huang, K.-T., "Query Optimization in Distributed Data Bases", MIT PhD thesis in preparation.

[ORE80]
Oren, T.I., C.M.Shub , and P.F.Roth, "Simulation with Discrete Models: A State-of-the-Art View", Volume 2, Proceedings of 1980 Winter Simulation Conference, Orlando Fl, Dec. 1980. 258 pages.

[SEG79]
Segall, A., "Optimal Distributed Routing for Virtual Line Switched- Data Networks", IEEE Transactions on Communications Vol COM-27, pp 201-209, Jan. 1979.

[TEN82]
Tenney, R.R. personal communication regarding ongoing student research efforts. Theses of one PhD student and two masters students nearing completion.

[WEL80]
Welch, J.A. "State of the Art in C2 Assessment", Proceedings for Quantitative Assessment of the Utility of Command and Control Systems, MITRE report MTR-80W00025, Washington, DC, Jan 1980, pp 11-20.

APPENDIX

A Sample TECCNET Session

```
++++TECCNET:

    Welcome to the Information Flow Network Testbed (TECCNET).
 For information on how to use the system, type:
          help
 followed by a carriage return.  Otherwise, on receiving the
 user cue (****USER::) you may type any TECCNET command.

****USER::
help

++++TECCNET:
     TECCNET is an interactive testbed which is intended to support
 the analysis of a number of information flow related issues.  It is
 structured to provide user support in three areas:

          1) the specification of his modeling environment, selection
             of local node models, message protocols and flow control
             strategies.  (program segment:  'model')

          2) the generation of simulation input and scenarios that define
             the network / traffic conditions. (program segment: 'scenario')

          3) the on-line execution of the simulation experiment (program
             segment:      'run').

    For information on how to interact with the TECCNET system, the
 inexpert user should type:        use

****USER::
use

++++TECCNET:
    You converse with TECCNET by entering commands and responses from your
 terminal.  When the user cue (****USER::) appears, it is the indication
 that you may begin typing.  To signal the computer that you have completed
 your entry you MUST strike the carriage return (CR) key.  The computer will
 not respond until you do.
     The commands permit you to move freely within the TECCNET system,
 to exercise the various programs, or to receive explanations and
 assistance in program selection and data preparation.  Occasionally,
 a specific response is required.  In these instances, you will be
 prompted from the terminal before the user cue is given.
     Depressing the BREAK or ATTN key at any time will interrupt TECCNET
 and return you to a point where you may again enter commands.
   The inexpert user should now type:
                    commands

****USER::
commands

++++TECCNET:
    The basic commands for controlling and interacting with the TECCNET
 system are organized into three groups:
```

```
        1) movement within the system:
           model: to change modeling environment
           scenario: to create scenario data
           run:   to execute the experiment
           stop:  to leave the system
        2) on-line documentation:
           help:  brief description of TECCNET
           use:   outline of interaction modes
           help model, help scenario, help run: descriptions of
                  the TECCNET submodels
           use model,use scenario, use run: instructions for their use
           help network, help traffic: description of input parameters
        3) user support:
           help error: how to correct errors in typing
           help comment: how to leave comments about the system
           comment: comment mechanism
           terse: request for brief interactive reporting
           verbose: request for complete interactive reporting

****USER::
help comment

++++TECCNET:
    The comment command is your way of recording comments throughout your
 TECCNET session.  Any information may be placed in a comment field (i.e.,
 difficulties with the system, documentation of the experiment being
 performed, etc).  If you type      comment        ,TECCNET will
 respond with a special comment prompt (++comment: ?).  The cursor will
 remain after the ? and you may begin your comment.  At the end of a line
 hit the carriage return to receive your next comment prompt and enter
 the next line of your message.  The format is free form, anything may
 be included in your text except the error correcting characters (@ and #).
 When you are through, merely hit carriage return immediately after
 receiving the prompt.

****USER::
comment

++++TECCNET:
 If you strike the carriage return immediately after receiving
 a comment prompt, you will terminate your message to us.

++comment ? This is a test case for a 5 node, 12 link network.
++comment ? Uniform capacities will be used on all links.
++comment ?

****USER::
rin

++++TECCNET:

 Your input cannot be interpreted as entered.
 If you need a list of active commands, type:      commands
 If you wish to leave the system; type:           stop
 Otherwise, retype your command and try again.

****USER::
help error

++++TECCNET:
```

```
    If you detect an error BEFORE you strike the carriage return key
 you may correct it in one of the following ways:
   1-Type the AT SIGN (@) to cancel the command line up to that point.
 Immediately retype the correct line.  Thus,if you type
         shop@stop
 you will get the 'stop' command correctly.
   2-n NUMBER SIGNS (#) will cancel the last n characters you typed.
 Thus, either of the two lines
      sh#top
      shop###top
 will enter the 'stop' command.  Combinations also work;e.g,
      shop###toa@show@stop

    If you wish to correct an error AFTER you have entered your
 command or to interrupt TECCNET, strike the BREAK or ATTN key
 and wait for the user cue.

****USER::
run

++++TECCNET:
 Your request to transfer to the 'run' portion of
TECCNET is being processed.

++++TECCNET:
    Please enter the name of the file in which your network
 data has been stored.  If no file exists or you wish to create
 a new file,  type:       new
 You will be prompted by the scenario builder for the data which defines
 your network.
 If you need clarification, type:      !help network

****USER::
net2

++++TECCNET:
    Please enter the name of the file in which your traffic
 data has been stored. If no file exists or you wish to create
 a new file,  type:          new
 You will be prompted by the scenario builder for the data which defines
 your traffic conditions.
 If you need clarification, type:      !help traffic

****USER::
terse

++++TECCNET:
 Your request for terse output is acknowledged.
 Traffic filename?

****USER::
traf2

++++TECCNET:
 Network and Traffic input complete. Would you like it displayed?
                   Answer yes or no

****USER::
yes
```

Network Topology

Link Name	Origin Node	Destination	Capacity
l1	n1	n2	100.
l2	n2	n1	100.
l3	n1	n5	100.
l4	n5	n1	100.
l5	n2	n3	100.
l6	n3	n2	100.
l7	n3	n4	100.
l8	n4	n3	100.
l9	n4	n5	100.
l10	n5	n4	100.
l11	n5	n2	100.
l12	n2	n5	100.

Desired Input Rates

Source	Destination	Desired rij
n1	n2	40.000
n1	n3	50.000
n1	n4	40.000
n2	n1	50.000
n2	n4	40.000
n5	n2	50.000
n5	n3	40.000
n3	n1	40.000
n3	n2	50.000
n3	n5	40.000
n4	n1	40.000
n4	n5	40.000
n4	n3	40.000

++++TECCNET:
Please indicate the desired number of iterations
Enter an integer >0

****USER::
50

SYSTEM SNAPSHOT at Time = .000

Objective function based on Ideal injections = 1122.748
on Effective injections = 1122.748

Network Injections

Source	Destination	Ideal rij	Effective rij
n1	n2	10.000	10.000
n1	n3	12.500	12.500
n1	n4	10.000	10.000
n2	n1	12.500	12.500
n2	n4	10.000	10.000
n5	n2	12.500	12.500
n5	n3	10.000	10.000
n3	n1	10.000	10.000
n3	n2	12.500	12.500
n3	n5	10.000	10.000

n4	n1	10.000	10.000
n4	n5	10.000	10.000
n4	n3	10.000	10.000

Link Flows

Link	Flow	Link	Flow	Link	Flow
l1	22.500	l2	22.500	l3	10.000
l4	10.000	l5	12.500	l6	32.500
l8	20.000	l9	20.000	l10	30.000
l11	12.500	l12	20.000		

++++TECCNET:
Do you wish to see updates in r and phi at individual nodes during the next iteration? Answer yes or no.

****USER::
no

SYSTEM SNAPSHOT at Time = 1.999

Objective function based on Ideal injections = 728.503
on Effective injections = 728.503

Network Injections

Source	Destination	Ideal rij	Effective rij
n1	n2	15.987	15.987
n1	n3	18.475	18.475
n1	n4	15.975	15.975
n2	n1	18.487	18.487
n2	n4	15.979	15.979
n5	n2	18.490	18.490
n5	n3	15.978	15.978
n3	n1	15.228	15.228
n3	n2	18.480	18.480
n3	n5	15.964	15.964
n4	n1	15.979	15.979
n4	n5	15.986	15.986
n4	n3	15.988	15.988

Link Flows

Link	Flow	Link	Flow	Link	Flow
l1	34.463	l2	33.716	l3	15.975
l4	15.979	l5	31.160	l6	49.674
l7	5.986	l8	25.269	l9	31.965
l10	35.249	l11	25.188	l12	25.956

++++TECCNET:
Do you wish to see updates in r and phi at individual nodes during the next iteration? Answer yes or no.

****USER::
no

SYSTEM SNAPSHOT at Time = 3.999

Objective function based on Ideal injections = 637.452
on Effective injections = 590.230

Network Injections

Source	Destination	Ideal rij	Effective rij
n1	n2	29.067	18.317
n1	n3	18.280	18.466
n1	n4	16.063	16.063
n2	n1	35.790	21.213
n2	n4	17.003	17.003
n5	n2	33.677	21.218
n5	n3	16.767	16.767
n3	n1	12.452	17.514
n3	n2	28.494	21.197
n3	n5	12.968	18.274
n4	n1	15.151	18.302
n4	n5	29.969	18.315
n4	n3	36.022	36.935

Link Flows

Link	Flow	Link	Flow	Link	Flow
l1	36.597	l2	48.157	l3	16.063
l4	17.194	l5	35.054	l6	51.462
l7	10.128	l8	44.101	l9	35.510
l10	33.060	l11	29.908	l12	25.584

++++TECCNET:
Do you wish to see updates in r and phi at individual nodes during the next iteration? Answer yes or no.

****USER::

++++TECCNET:
You may now enter any TECCNET command.

****USER::
comment

++++TECCNET:
If you strike the carriage return immediately after receiving a comment prompt, you will terminate your message to us.

++comment ? should add additional links and change the experiment
++comment ? _back to a definition stage.
++comment ?

****USER::
stop

++++TECCNET:
You are now leaving the TECCNET system.